Errors in numerical analysis

Numerical analysis - University of Luxembourg - Session 1

Exercises

Python documentation

- numpy: https://numpy.org/doc/
- matplotlib: https://matplotlib.org/stable/gallery/subplots_axes_and_figures/index.html
- sympy https://www.sympy.org/en/index.html

Exercises on numerical errors

Exercise 1. **Absolute and relative errors**

Step 1: Print the relative and absolute errors of the Stirling approximation

$$n! \approx \sqrt{2\pi n}\left(\frac{n}{e}\right)^n$$

for different values of n. Explain your observations for large values.

- Implement a function that computes the Stirling approximation "def Stirling(n): return ...", and another one that computes the errors "def Errors(x, xref): ... return abs err, rel err"
- For factorials, you can use “from scipy.special import factorial”
- For simple plotting, you can use the following concise form
    - plt.figure(figsize=(7,4))
    - plt.plot(N, err0, 'm-o', label = 'Stirling') or plt.loglog(N, err0, 'm-o', label = 'Stirling')

Step 2: Do again the comparison with more terms of the asymptotic development

$$n! \approx \sqrt{2\pi n}\left(\frac{n}{e}\right)^n [1 + \frac{1}{12n} + \frac{1}{288n^2} - \frac{139}{51840n^3} + O(\frac{1}{n^4})]$$

- You can implement more functions def Stirling2(n) and def Stirling3(n):

Comment on the accuracy of the approximation.

Exercise 2. **Round-off vs discretization error**

Let $f(x) = sin(x)$ and $x_0 = 1.2$.

1. Write a Taylor expansion of f around x_0 with a step size h, and derive a formula to approximate $f'(x_0)$ of first order $O(h)$ (see the lecture slides).

2. Plot the absolute error in a log-log scale as a function of the step size h, where h is in the range $[10^{-16}, 1]$. What is going on when h is too small?

3. Add another plot by changing the numpy floating point precision using np.float32, for both x_0 and h. Please comment on your observations

- use np.logspace to generate the h values

Exercises on floating point systems

Exercise 3. **Decimal approximations**

Compute decimal approximations of the real number 0.1 for 32-bit floating number precision in base 2. Then do the same thing for 0.25. Explain the difference that you observe. What about 0.35?

- You can use f-strings for printing the values print(f'\nSingle Precision (32-bit): {single_precision:.20f}')

Exercise 4. **Decimal Conversion of a Single-Precision Floating Point Number**

Find the decimal equivalent of the following 32-bit precision machine number (sign, exponent, significand):

0 10000000 10010010000111111011011

- Mantissa 1's are in position 1,4,7,12-17,19-20, 22-23

This decimal number approximates a well-known number. How many significant digits does this 32-bit representation achieve?

Exercises on stability and conditioning

Exercise 5. **Polynomials**

Evaluate and plot the polynomial function $(1 - x)^6$ written in its current factored and developed forms. Look in particular at the behavior close to the root $x = 1$. You can use the interval $I = [0.995, 1.005]$ equally spaced by 100 points. Explain your observations.

- One can use sympy to find expression for the developed form

Exercise 6. **Catastrophic cancellation**

Recall the quadratic formula to find the root of $ax^2 + bx + c$. Test the usual formula for the coefficients. $a = 1,\ b = -(10^8 + 10^{-8}), c = 1$. Explain what is going on, and propose a method to compute both roots accurately.